

[INICIAR SESIÓN](#)[NUESTROS PLANES](#)[TODOS LOS CURSOS](#)[FORMACIONES](#)[CURSOS](#)[PARA EMPRESAS](#)[ARTÍCULOS DE TECNOLOGÍA > FRONT END](#)

Crear una máscara de Teléfono con Javascript



Felipe Nascimento

26/09/2020

Una editora de libros tiene en su sitio una página de hable con nosotros para atender mejor a sus clientes. En esta página se creó un formulario con nombre, dirección y teléfono. El problema es que, el campo de teléfono está sin formato permitiendo que el cliente ingrese los números que quiere. Con eso el campo puede recibir un número como:

9999999999999999

Y sería difícil saber si ese número es un teléfono fijo o un móvil. Es decir, es confuso ya sea para el cliente que está ingresando el teléfono en el campo, o para el backend que recibe esta información. El formato esperado para teléfonos móviles, por ejemplo sería:

99999-9999

Entonces, lo que vamos a hacer es encontrar una manera de solucionar nuestro problema y llegar al resultado esperado utilizando JavaScript.

Capturando el valor del input

El tramo del html donde colocamos el número de teléfono está de la siguiente manera:

```
<div>
  <input type="text" placeholder="9999-9999 o 9999-99999" />
</div>
```

Lo que queremos ahora es capturar el número digitado por el cliente, entonces vamos a crear una variable para que nos ayude en esta tarea:

- El `textoActual` es el responsable por capturar el número del teléfono.

Vamos a aprovechar y encapsular nuestro código dentro de una función, pensando en facilitar la legibilidad y futuros mantenimientos en nuestro código.

```
function mascaraDeTelefono( telefono ){
  const textoActual = telefono.value;
}
```

Ahora ya tenemos el `value` del `input`, falta encontrar una manera de decir si el número de teléfono es fijo o móvil.

Teléfono fijo o móvil

Desarrollaremos una lógica que consistirá en un `if else` que confirmará el tipo de teléfono y luego pasaremos este número al variable `telefono` ya formateado.

```
function mascaraDeTelefono(telefono){
  const textoActual = telefono.value;
  const isCelular = textoActual.length === 9;
  let textoAjustado;
  if(isCelular) {
    // hace alguna cosa
  } else {
    // hace alguna cosa
  }
  telefono.value = textoAjustado;
}
```

Ahora lo que queremos es una manera de formatear estos números en el siguiente estándar: XXXXX-XXXX para móvil y XXXX-XXXX para teléfono fijo.

Eliminando el Guión

Primero eliminaremos el guión del número ingresado, utilizando [el método replace](#) que devolverá una nueva string, con un nuevo estándar preestablecido.

```
function eliminarGuion(teléfono) {  
    const textoActual = teléfono.value;  
    const textoAjustado = textoActual.replace(/\-/g, '');  
  
    telefono.value = textoAjustado;  
}
```

Método slice

Como el input que capturamos es una string, y ya eliminamos el guión del número, una posible solución para nuestro problema de formateo es contar cuantos caracteres tienen esta string y enseguida cortarla en dos partes - una antes y otra después del guión. Una manera de cortar la string es por el [método slice](#).

El `slice` nos permite manipular arrays y strings (son arrays de caracteres), devolviendo solamente el tramo que elegimos.


Cómo dividiremos la secuencia de números en dos partes, creamos dos variables que recibirán estas partes y otra que será responsable por formatear los números:

- `const parte1`, que tomará la primera posición hasta la posición 5
- `const parte2`, que tomará de la posición 5 hasta la última posición
- `textoAjustado`, que interpolará el resultado de las dos variables con el guión

```
let textoAjustado;  
if(isCelular) {  
    const parte1 = textoActual.slice(0,5);  
    const parte2 = textoActual.slice(5,9);  
    textoAjustado = `${parte1}-${parte2}`  
} else {
```

```
    const parte1 = textoActual.slice(0,4);  
    const parte2 = textoActual.slice(4,8);  
    textoAjustado = `${parte1}-${parte2}`  
  }  
  
  telefono.value = textoAjustado;  
}
```

Con el código listo tenemos el siguiente resultado:



99999-9999

¡Excelente! Conseguimos formatear como queríamos. Ahora, con lógica lista necesitamos encontrar una manera de disparar esta función todas las veces que el cliente haya ingresado el teléfono en el campo.

El evento onblur

El [evento blur](#) se activa cuando un elemento pierde el foco, es decir, después que el cliente haya ingresado el teléfono y pulse en otro campo, el evento onblur se dispara accionando nuestra función y realizando el formateo del teléfono.

Pasamos el `this` como parámetro de la función para decir que el parámetro está dentro del contexto de nuestra función.

```
<div>  
  <input type="text" onblur="mascaraDeTelefono(this)"  
    placeholder="9999-9999 o 9999-99999"/>  
</div>
```

El evento onfocus

Con la función lista, la uniremos al [evento onfocus](#) que será responsable por disparar nuestra función, mientras el campo del input esté en enfocado.

```
<div>
  <input type="text" onblur="mascaraDeTelefono(this)"
    placeholder="9999-9999 o 9999-99999"
    onfocus="tiraHifen(this)" />
</div>
```

Como vimos, las validaciones son siempre problemas complejos en virtud de las innúmeras posibilidades de interacción del usuario.

Para saber más

Otra posibilidad de hacer nuestro formateo sería a través de las [expresiones regulares](#), que no son más que combinaciones de caracteres que utilizamos para seleccionar otros caracteres en strings.

En nuestro caso, podríamos utilizar la siguiente expresión regular para formatear el número del móvil, por ejemplo:

```
let textoAjustado;
if(isCelular) {
  textoAjustado = textoActual.replace(/(\d{5})(\d{4})/,
    function( regex, arg1, arg2) {
      return arg1 + '-' + arg2 ;
    });
  telefono.value = textoAjustado;
}
```

Si te interesa este tema, aquí en [Alura](#) tenemos capacitaciones de front-end, para principiantes y para aquellos que ya son desarrolladores web. En ellas, verás cómo programar en Javascript, HTML y CSS para construir sitios web.

ARTÍCULOS DE TECNOLOGÍA > FRONT END

En Alura encontrarás variados cursos sobre Front End. ¡Comienza ahora!

SEMESTRAL

US\$49,90

un solo pago de US\$49,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

ANUAL

US\$79,90

un solo pago de US\$79,90

- ✓ 218 cursos
- ✓ Videos y actividades 100% en Español
- ✓ Certificado de participación
- ✓ Estudia las 24 horas, los 7 días de la semana
- ✓ Foro y comunidad exclusiva para resolver tus dudas
- ✓ Acceso a todo el contenido de la plataforma por 12 meses

¡QUIERO EMPEZAR A ESTUDIAR!

[Paga en moneda local en los siguientes países](#)

Acceso a todos
los cursos

Estudia las 24 horas,
dónde y cuándo quieras

Nuevos cursos
cada semana

NAVEGACIÓN

PLANES

INSTRUCTORES

BLOG

POLÍTICA DE PRIVACIDAD

TÉRMINOS DE USO

SOBRE NOSOTROS

PREGUNTAS FRECUENTES

¡CONTÁCTANOS!

¡QUIERO ENTRAR EN CONTACTO!

BLOG

PROGRAMACIÓN

FRONT END

DATA SCIENCE

INNOVACIÓN Y GESTIÓN

DEVOPS

AOVS Sistemas de Informática S.A

CNPJ 05.555.382/0001-33

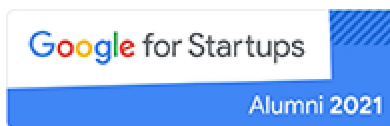
SÍGUENOS EN NUESTRAS REDES SOCIALES



ALIADOS



En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.



Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación

Lógica de Programación | Java

Cursos de Front End

HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics |

Liderazgo y Gestión de Equipos | Startups y Emprendimiento